

Figure 1A

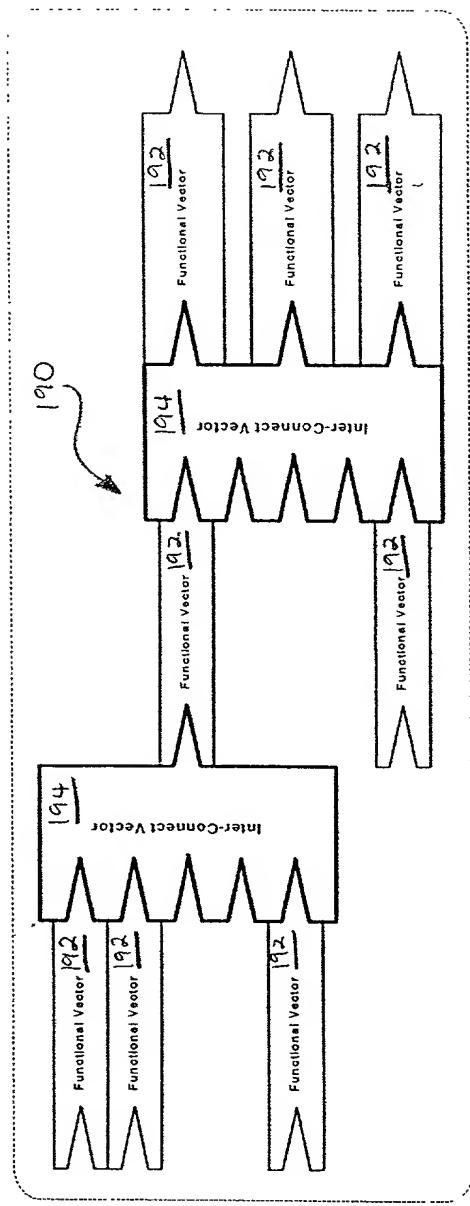


Figure 1B

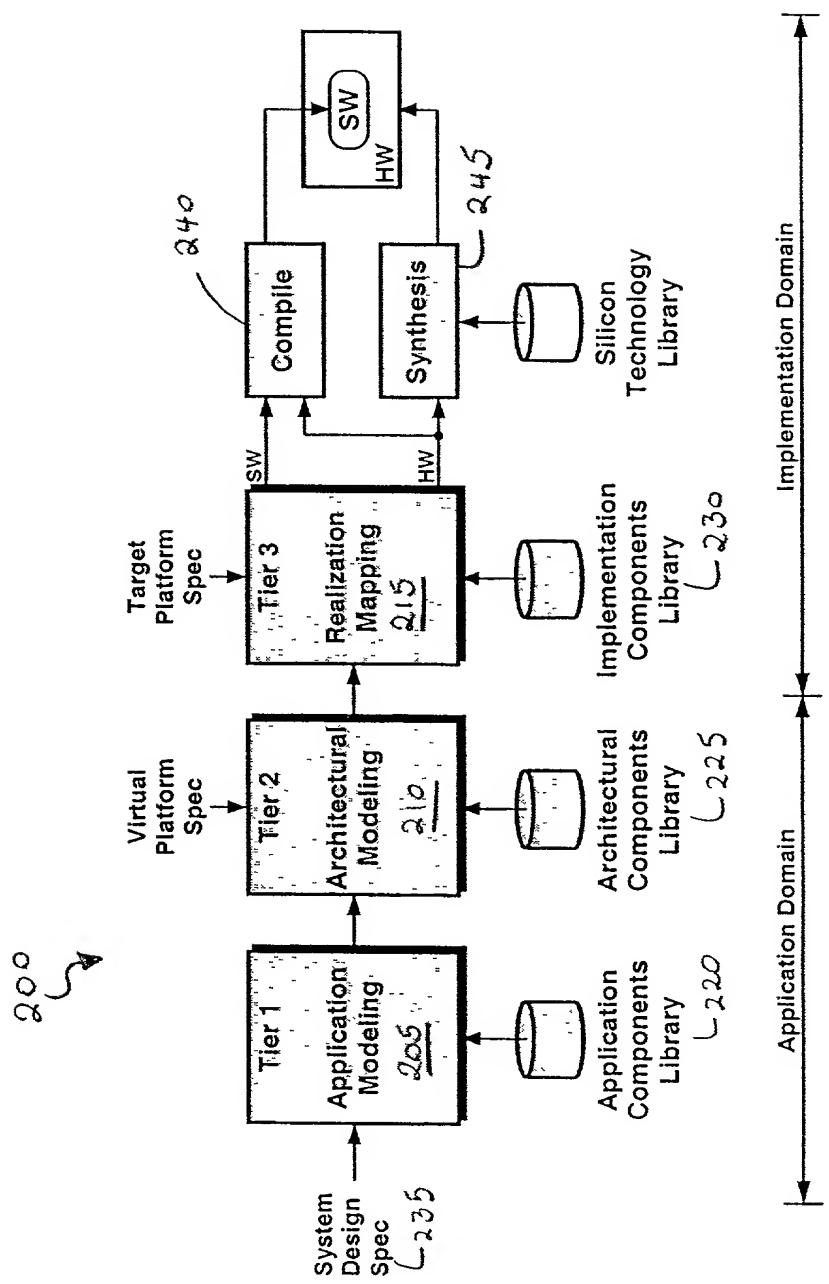


Figure 2

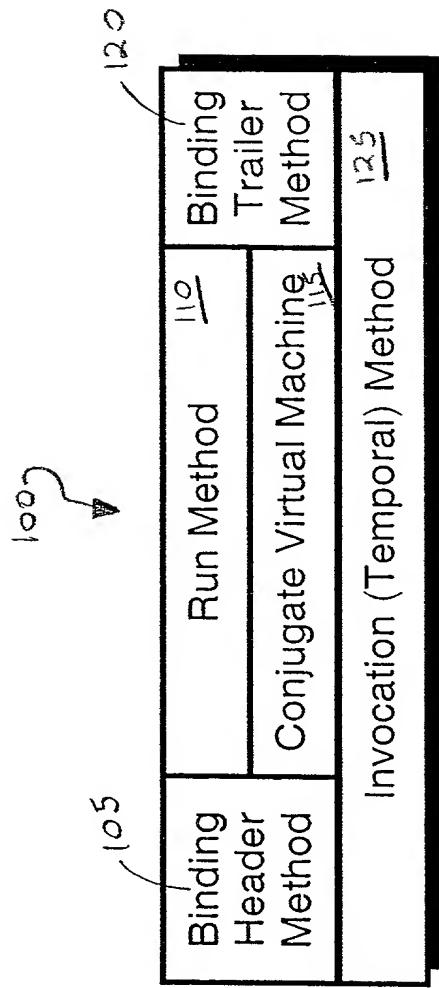


Figure 3A

130
↓

```
/** Vector Attributes */  
string vectorName; 135  
string vectorType; 140  
string parentAS; 145
```

Figure 3B

150
↓

```
/** Header variables */  
// Add input variable declarations  
Object headerVar[]; 155  
/** Trailer variables */  
// Add output variable declarations  
Object trailerVar[]; 160
```

Figure 3C

170

172

```

    /** Vector Constructor Method: Construct an actor with the given vector name */
    public udmVectorClassName (string vectorName, udmVector inVector[], udmVector
    outVector[])
    {
        // Call constructor in base class
        super(vectorName, parentAS, inVector, outVector);
        // Perform any initialization that needs to be done in the constructor
    }

    /** This method contains the actual behavior of the vector */
    private boolean vectorRun() 174
    {
        // Perform the vector processing
        return true; // (or false if you want to terminate the thread)
    }

    /** This is the invocation method that checks to see if the vector is ready to run */
    private void vectorInvocation() 176
    {
        while ( !headerDataReady() ) vectorWait();
    }

    /** Get the header input data */
    private void getHeaderInput() 178
    {
        // Get input data from interconnect vector
        headerData = vectorGet();
    }

    /** Send the trailer output data */
    private void sendTrailerOutput() 180
    {
        // Send output data to the interconnect vector
        vectorSend( trailerData );
    }

    /** run is the method that is started by Java when the thread is started */
    public void run() 182
    {
        boolean runThread = true;

        // Initialize the vector
        initialize();
        while ( runThread )
        {
            // Call invocation method
            vectorInvocation();

            // Get input data
            getHeaderInput();

            // Do the processing for the vector
            runThread = vectorRun();

            // Send output data
            sendTrailerOutput();
        }

        // Perform final cleanup before vector thread exits
        wrapup();
    }
}

```

Figure 3D

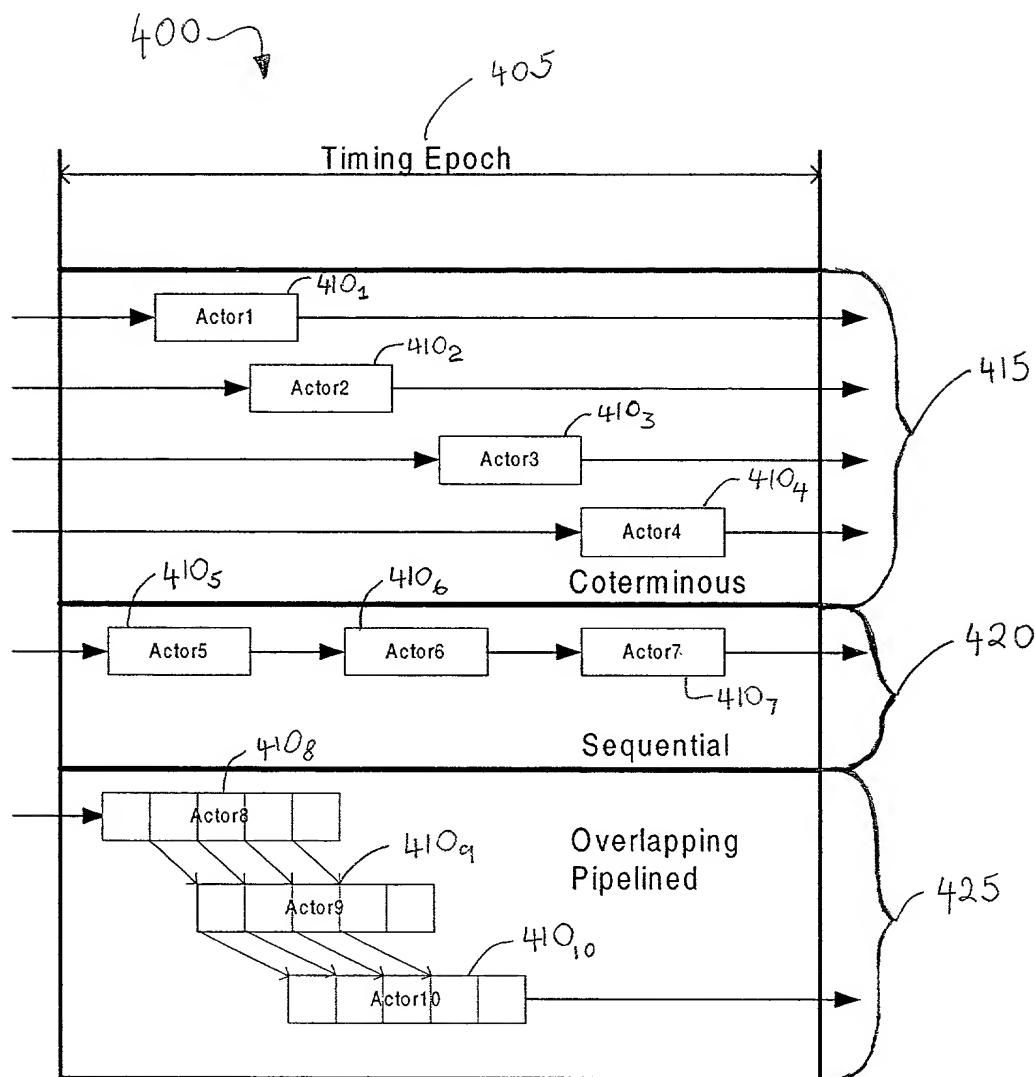


Figure 4

Application Syntax Structural Model

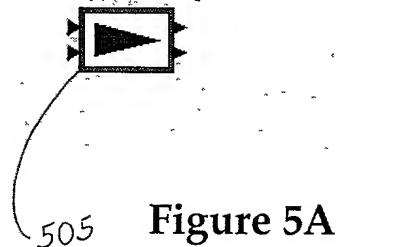


Figure 5A

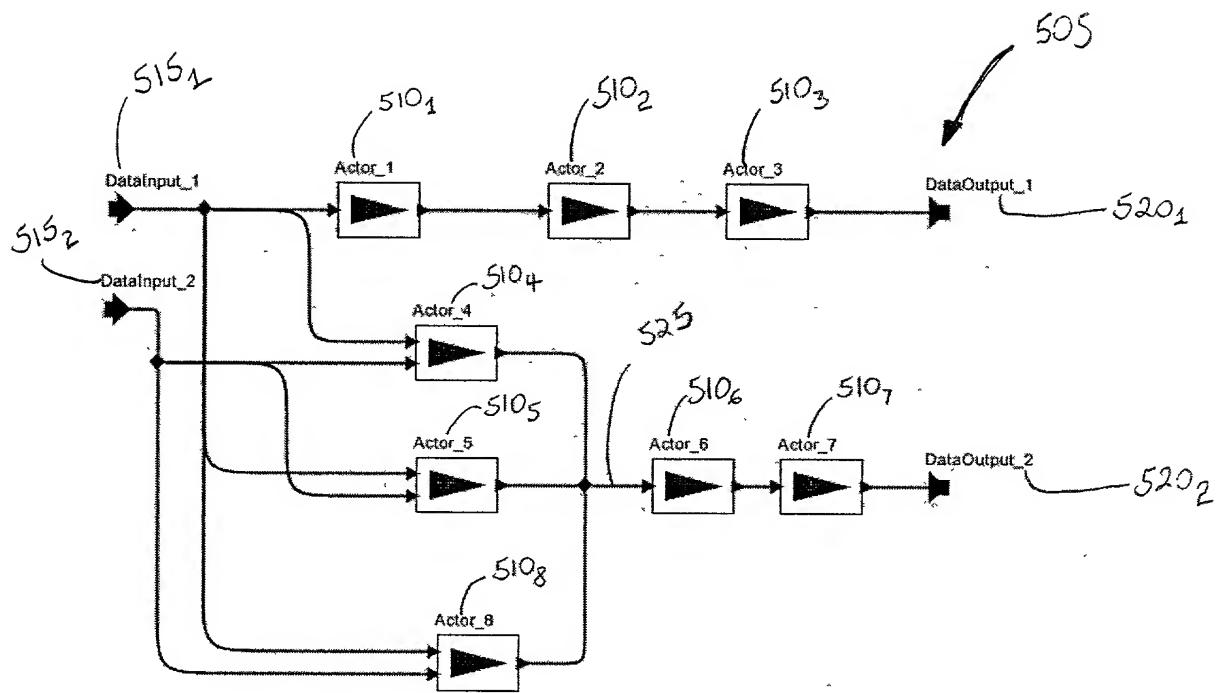


Figure 5B

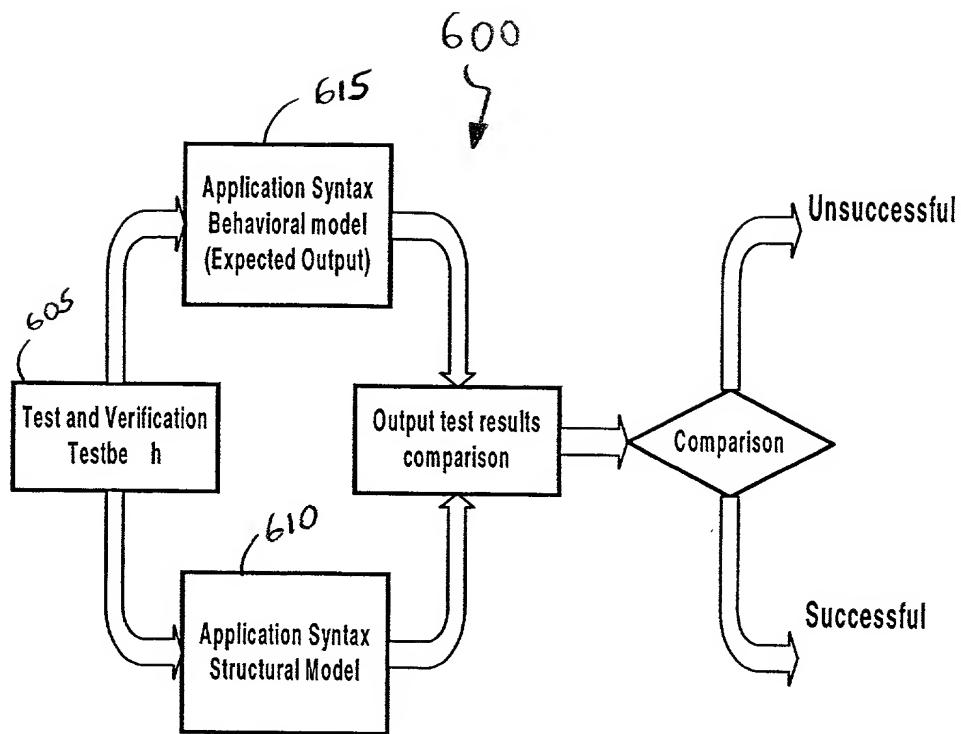


Figure 6